

# ED: Factor, Array, List, and Dataframe



# Factor

- A factor is a vector that is used to specify a discrete classification of the components of other vectors of the same length.

```
> students.origin=c("londres", "paris", "madrid", "madrid", "paris", "roma")  
> fstudents=as.factor(students.origin)
```

```
> fstudents  
[1] londres paris  madrid madrid paris  roma  
Levels: londres madrid paris roma
```

```
> summary(fstudents)  
londres  madrid  paris   roma  
      1      2      2      1
```

# Factor

- To know the name of the levels:

```
Data<-factor(c("mujer", "hombre", "mujer"))
```

```
> levels(Data)  
[1] "hombre" "mujer"  
> nlevels(Data)  
[1] 2
```

- By default, factor levels are treated in alphabetical order

# Ordered Factors: ordered()

- They are factors whose levels keep a certain order.
- Consider the cholesterol level of 10 patients:

```
>nivel.col=c("medio","medio","bajo","medio","bajo","medio","alto","alto",  
"bajo","bajo")
```

```
>nivel.col.ord=ordered(nivel.col,levels=c("bajo","medio","alto"))
```

```
> nivel.col.ord
```

```
[1] medio medio bajo medio bajo medio alto alto bajo bajo
```

```
Levels: bajo < medio < alto
```

- If you want to know the patients with a given cholesterol level:

```
> nivel.col.ord<"alto"
```

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE  
TRUE TRUE
```

```
> nivel.col.ord<"medio"
```

```
[1] FALSE FALSE TRUE FALSE TRUE FALSE FALSE FALSE  
TRUE TRUE
```

# Ordered Factors: ordered()

- **With vector:**

```
>nivel2.col=c("medio","medio","bajo","medio","bajo","medio","alto","alto",  
", "bajo","bajo")  
> nivel2.col  
[1] "medio" "medio" "bajo" "medio" "bajo" "medio" "alto" "alto" "bajo"  
"bajo"
```

```
> nivel2.col<"alto"  
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
FALSE FALSE
```

- **With factor:**

```
> nivel.col.ord<"alto"  
[1] TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE TRUE  
TRUE
```

# Array

- Array: generalization of a matrix to the multidimensional case.

```
array(data,dimensions)
```

# Array

`array(1:12,c(2,3,2))`

`x=array(c(75,72,65,70,85,85,55,60,75,70,80,80),c(2,3,2) )`

`dimnames(x) =  
list(c("hombres","mujeres"),c("Estadistica","Fisica",  
"Programacion"), c("getafe", "leganes"))`

```
, , 1
  [,1] [,2] [,3]
[1,]  1   3   5
[2,]  2   4   6

, , 2
  [,1] [,2] [,3]
[1,]  7   9  11
[2,]  8  10  12
```

```
, , getafe
             Estadistica  Fisica  Programacion
hombres      75          65          85
mujeres      72          70          85

, , leganes
             Estadistica  Fisica  Programacion
hombres      55          75          80
Mujeres      60          70          80
```

# Array

- To select elements of a array you have to use positions of element

“[“ + dim1+ “, “+ dim2 + .... “, “+ dim-n.... + ”]”

**x[,, "leganes"]**

	Estadistica	Fisica	Programacion
hombres	55	75	80
mujeres	60	70	80

**x["hombres",,]**

	getafe	leganes
Estadistica	75	55
Fisica	65	75
Programacion	85	80



# List

- To concatenate objects where each element can have a different structure.
- A list has components, to which a name must be assigned.

```
> student=list(nia="1000001",name="Francisco Sanz  
Perez",dni="44544432D")
```

```
> student
```

```
$nia
```

```
[1] "1000001"
```

```
$name
```

```
[1] "Francisco Sanz Perez"
```

```
$dni
```

```
[1] "44544432D"
```

# List

- To visualize the names of the object in the list

```
> names(student)
[1] "nia"  "name" "dni"
```

- To access specific components, you can use:
  - \$ followed by the name of the component
  - [[component number]]

```
> alumno$dni
[1] "44544432D"
```

```
> alumno[[3]]
[1] "44544432D"
```

# Data Frames

- Data Frames: data structure that generalizes to the matrices:
  - the columns can be of different types from each other
  - elements of the same column must be of the same type
  - elements must be of the same length.

# Data Frames

```
data=matrix(c(7.5,7.2,6.5,7.0,8.5,8.5,5  
.5,6.0,7.5,7.0,8.0,8.0),nrow=6,byrow=  
T)  
> datos
```

```
>data  
      [,1] [,2]  
[1,] 7.5  7.2  
[2,] 6.5  7.0  
[3,] 8.5  8.5  
[4,] 5.5  6.0  
[5,] 7.5  7.0  
[6,] 8.0  8.0
```

```
>dimnames(data)=list(c("ana","pepe","  
nacho","bea","gema","alba"),  
c("Matematicas","Fisica"))
```

```
> data
```

	Matematicas	Fisica
ana	7.5	7.2
pepe	6.5	7.0
nacho	8.5	8.5
bea	5.5	6.0
gema	7.5	7.0
alba	8.0	8.0

# Data Frames

```
>provincia=c("madrid","leon","oviedo",  
,"malaga","sevilla", "madrid")  
> data2=cbind(data,provincia)
```

```
> data2
```

	Matematicas	Fisica	provincia
ana	"7.5"	"7.2"	"madrid"
pepe	"6.5"	"7"	"leon"
nacho	"8.5"	"8.5"	"oviedo"
bea	"5.5"	"6"	"malaga"
gema	"7.5"	"7"	"sevilla"
alba	"8"	"8"	"madrid"

```
> mean(data["Matematicas"])  
[1] 7.25
```

```
> mean(data2["Matematicas"])  
[1] NA
```

Mensajes de aviso perdidos

In mean.default(data2[, "Matematicas"]) :  
argument is not numeric or logical:  
returning NA

# Data Frames

```
data2=data.frame(datos,provincia)
```

	Matematicas	Fisica	provincia
ana	7.5	7.2	madrid
pepe	6.5	7.0	leon
Nacho	8.5	8.5	oviedo
bea	5.5	6.0	malaga
gema	7.5	7.0	sevilla
alba	8.0	8.0	madrid

```
> mean(data2[, "Matematicas"])  
[1] 7.25
```

```
# To access the data  
> data2[, "Fisica"]  
[1] 7.2 7.0 8.5 6.0 7.0 8.0
```

```
> data2$Fisica  
[1] 7.2 7.0 8.5 6.0 7.0 8.0
```